Non-Full-Active Super-Sbox Analysis: Applications to ECHO and Grøstl

Yu Sasaki[1], Yang Li[2], Lei Wang[2], Kazuo Sakiyama[2], and Kazuo Ohta[2]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho, Musashino-shi, Tokyo 180-8585 Japan
sasaki.yu@lab.ntt.co.jp
[2] The University of Electro-Communications
1-5-1 Choufugaoka, Choufu-shi, Tokyo, 182-8585 Japan
{liyang,wanglei,saki,ota}@ice.uec.ac.jp

Abstract. In this paper, we present *non-full-active Super-Sbox analysis* which can detect non-ideal properties of a class of AES-based permutations with a low complexity. We apply this framework to SHA-3 round-2 candidates ECHO and Grøstl. The first application is for the full-round (8-round) ECHO permutation, which is a building block for 256-bit and 224-bit output sizes. By combining several observations specific to ECHO, our attack detects a non-ideal property with a time complexity of 2^{182} and 2^{37} amount of memory. The complexity, especially in terms of the product of time and memory, is drastically reduced from the previous best attack which required $2^{512} \times 2^{512}$. Note that this result does not impact the security of the ECHO compression function nor the overall hash function. We also show that our method can detect non-ideal properties of the 8-round Grøstl-256 permutation with a practical complexity, and finally show that our approach improves a semi-free-start collision attack on the 7-round Grøstl-512 compression function. Our approach is based on a series of attacks on AES-based hash functions such as rebound attack and Super-Sbox analysis. The core idea is using a new differential path consisting of only non-full-active states.

Keywords: AES-based permutation, ECHO, Grøstl, SHA-3, Super-Sbox

1 Introduction

Hash functions are used in the wide range of cryptographic applications. Since the break of MD5 and SHA-1 [1, 2], cryptographers have been seeking secure and efficient hash constructions. From these backgrounds, NIST started the competition to determine the future standard hash function called SHA-3 [3].

In the SHA-3 competition, 14 algorithms are being considered as round 2 candidates. At the present time, none of them has been seriously broken in terms of the important security properties such as collision resistance or preimage resistance. However, regarding some candidates, building blocks such as compression functions or internal permutations have been shown that they do not satisfy

ideal properties. Although it does not damage the security of hash functions immediately, the analyses against building blocks are useful to know the potential weakness, security margin, validity of the security proof, and so on.

Many of the SHA-3 candidates are based on the design strategy of AES [4, 5]. Recently, an outstanding progress in the cryptanalysis against AES-based hash functions or permutations has been made [6–16]. Specifically, *Rebound attack* proposed by Mendel *et al.* at FSE 2009 [7], *Start-from-the-Middle attack* proposed by Mendel *et al.* at SAC 2009 [8], and *Super-Sbox analysis* applied to the rebound attack by Lamberger *et al.* at Asiacrypt 2009 [15] and by Gilbert and Peyrin at FSE 2010 [9] have wide range of their applications and are powerful analytic tools. In fact, the rebound based attack has been applied to several SHA-3-candidates [7–14, 17] such as Grøstl [18], ECHO [19], JH [20], Cheetah [21], LANE [22], Twister [23]. It has also been applied to other hash functions [7–9, 15, 16] such as Whirlpool [24] and AES hashing modes.

ECHO [19], designed by Benadjila *et al.*, is one of the round 2 algorithms in the SHA-3 competition using a 2048-bit AES-based permutation. The number of rounds in the permutation is 8 for ECHO-224 and -256, and 10 for ECHO-384 and -512. At FSE 2010, Gilbert and Peyrin showed that the full-round (8-round) ECHO permutation could be distinguished from an ideal permutation with time of 2^{768} and memory of 2^{512} by using the Super-Sbox analysis [9]. After that, Peyrin [25, 26] improved this attack which required 2^{512} in both time and memory. Because the 8-round ECHO permutation is a building block to generate 256-bit or 224-bit hash values and compression part from 2048-bits to 256- or 224-bits is not considered, the impact of this attack seems almost negligible. In addition, as long as it is evaluated by the framework of [9], the time or memory cannot be below 2^{512} [1]. To sum up, there is no powerful analysis on the ECHO hash function nor compression function. Even though attacks on the permutation reached full-round, the complexity is too high.

Note that the reduced ECHO compression function is attack by Peyrin [26]. Recently, Schläffer presented the analysis on ECHO [27] and Ideguchi *et al.* presented the analysis on Grøstl [28]. These results are listed in Table 1.

Our Contributions

In this paper, we present *non-full-active Super-Sbox analysis* which can detect non-ideal properties of a class of AES-based permutations with a low complexity. To demonstrate its applicability, we first apply the non-full-active Super-Sbox analysis to the 8-round Grøstl-256 permutation, which is an AES-based permutation consisting of the 8×8 state. This attack can detect a non-ideal property of the 8-round Grøstl-256 permutation with time of 2^{48} and memory of 2^8, while detecting the same property of an ideal permutation requires 2^{96}. We then apply this framework to the full-round (8-round) ECHO permutation by optimizing the attack with taking several properties specific to ECHO into account. This attack can detect a non-ideal property of the 8-round ECHO permutation with time of

[1] Reasons of this limitation are explained in [9, Section 4.4] and [26, Appendix B].

Table 1. Comparison of attack results on ECHO and on Grøstl.

Target	Rounds	Time	Memory	Attack Type	Paper
ECHO-256/-224 Permutation	8 (full)	2^{768}	2^{512}	Distinguisher	[9]
	8 (full)	2^{512}	2^{512}	Distinguisher	[26]
	8 (full)	2^{182}	2^{37}	Distinguisher	Sect. 5.2
	7	2^{128}	2^{32}	Distinguisher	[26]
	7	2^{118}	2^{38}	Distinguisher	Append. A
ECHO-256/-224 Single-pipe Comp. Func.	3	2^{64}	2^{64}	Distinguisher	[26]
	3	2^{32}	2^{38}	Distinguisher	Append. B
Grøstl-256 Permutation	8	2^{112}	2^{64}	Distinguisher	[9]
	8	2^{64}	2^{64}	Distinguisher	[28]
	8	2^{48}	2^{8}	Distinguisher	Sect. 4.4
Grøstl-512 Comp. Function	7	2^{152}	2^{64}	Semi-free-start coll.	[17]
	7	2^{152}	2^{56}	Semi-free-start coll.	Sect. 5.3
ECHO-256 Hash Function	4	2^{64}	2^{64}	Collision	[27]
	5	2^{96}	2^{64}	Distinguisher	[27]
ECHO-256 / -512 Comp. Function	3/3	$2^{64}/2^{96}$	$2^{64}/2^{64}$	Semi-free-start coll.	[26]
	7/7	$2^{107}/2^{106}$	$2^{64}/2^{64}$	Distinguisher	[27]
Grøstl-256 Comp. Func.	10 (full)	2^{192}	2^{64}	Distinguisher	[26]
Grøstl-512 Comp. Func.	11	2^{640}	2^{64}	Distinguisher	[26]

2^{182} and memory of 2^{37}, while detecting the same property of an ideal permutation requires 2^{256}. Note that the 8-round ECHO permutation is a building block for ECHO-256 and ECHO-224. As far as we know, this is the first result on the full-round ECHO permutation which can work with both time and memory (or product of these factors) below 2^{256} (or 2^{224}). Note, however, that the role of the convolution in the ECHO compression function is very important for its security and our distinguisher cannot be extended to the ECHO compression function, nor the hash function. Finally, we show that our approach also improves the amount of memory for the semi-free-start collision attack on the 7-round Grøstl-512 compression function to 2^{56} from 2^{64}. In appendices, we show new results on the reduced-round ECHO permutation and compression function. An attack on the 7-round ECHO permutation and a low complexity distinguisher on the 3-round single-pipe ECHO-256 compression function are included. The attack results are summarized in Table 1. The technical details in this paper are as follows.

Low complexity distinguishers on AES-based permutations We present a new strategy of the Super-Sbox analysis which can work for a class of AES-based permutations in generic. The core idea is using a differential path whose inbound part, in particular inside the Super-Sbox, consists of only non-full-active states. Regarding non-active bytes, the difference is al-

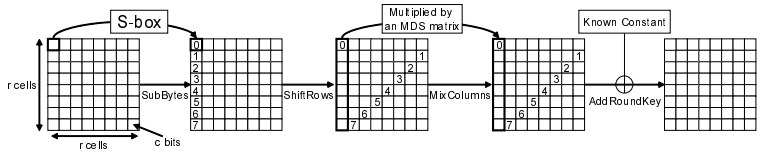

Fig. 1. The operations inside a round of AES-based permutation.

ways 0 through the SubBytes and InverseSubBytes operations regardless of its value. Hence, attackers can freely choose the value without breaking the differential path. This freedom degrees enable attackers to control values (or differences through the SubBytes operation) of other bytes inside the Super-Sbox to be connected efficiently.

Observations on the property of ECHO permutation We explain two new observations on the ECHO permutation when dealing with the byte-wise truncated differential path. First, we find that the linearity of the jointed two linear operations (MixColumns inside the BigSB and the following BigMC) should be taken into account in order to correctly calculate the complexity for a certain differential path. Second, there are freedom of the differential paths inside BigSB available to attackers to reduce the complexity.

In Section 2, we describe AES-permutation, ECHO, and Grøstl. In Section 3, we introduce previous work. In Section 4, we present the framework of non-full-active Super-Sbox analysis and show its application to the 8-round Grøstl-256 permutation. In Section 5, we attack the full-round ECHO permutation and the 7-round Grøstl-512 compression function. In Section 6, we conclude this paper. Results on other variants of ECHO are described in appendices.

2 Specifications

AES [4, 5] is a 128-bit block-cipher represented by a 4×4 byte state. Here we consider a general AES-based permutation with $r \times r$ state where each element is a c-bit word. The row and column positions of a word/byte is denoted by i and j, respectively where $i, j \in [0, r-1]$. As shown in Fig. 1, the state is updated by four operations in a round of the AES-based permutation.

- SubBytes (SB): non-linear word/byte substitution according to an S-box.
- ShiftRows (SR): each word/byte at row j is rotated to left by j positions.
- MixColumns (MC): multiply each column by a MDS matrix.
- AddRoundKey (AK): bit-wise XOR of the current state and a constant.

2.1 ECHO Permutation

ECHO [19] designed by Benadjila *et al.* is a hash function using a 2048-bit AES-based permutation as its building block. The permutation consists of 8 rounds

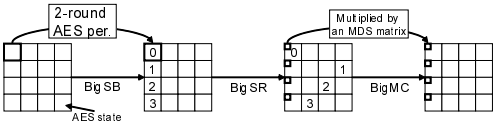

Fig. 2. One round of ECHO permutation.

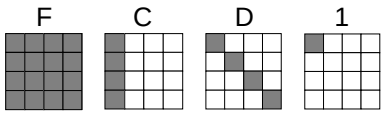

Fig. 3. Notations for ECHO BigWords.

for ECHO-224 and -256, and 10-rounds for ECHO-384 and -512. The 2048-bit internal state can be represented by a 4×4 matrix where each element is a 128-bit AES state called a BigWord. The round operation in the ECHO permutation manipulates 128-bit BigWords instead of 8-bit bytes. One round of the ECHO permutation shown in Fig. 2 has three operations:

- BigSB: substituting each BigWord by applying two AES-rounds.
- BigSR: each BigWord at row j is rotated to left by j positions.
- BigMC: multiply each 4 bytes of the ECHO state by a MDS matrix.

To simplify the dedicated analysis on ECHO, as introduced by [26], we denote 4 types of byte-wise truncated differences of the BigWord as shown in Fig. 3.

2.2 Grøstl Permutation and Compression Function

Grøstl designed by Gauravaram *et al.* [18] is another hash function built upon the AES-based permutations. Grøstl-256 permutation uses an 8×8 state where each element is an 8-bit byte, while Grøstl-512 permutation uses an 8×16 state. The number of rounds in the permutation is 10 for Grøstl-224 and -256, and 14 for Grøstl-384 and -512. The Grøstl-512 uses different ShiftRows operation from Grøstl-256, where the bytes at row 7 are rotated to left by 11 positions.

3 Previous Work

Rebound attack was proposed by Mendel *et al.* at FSE 2009 [7], which is useful to analyze AES-based permutations. It divides a differential path into two parts; inbound and outbound phases. Inbound phase controls the most expensive part of the differential path with a very low average complexity, then outbound path is satisfied probabilistically. It needs to make sure the total number of starting points generated at the inbound phase is enough to fulfill the outbound path.

Start-from-the-Middle attack was proposed by Mendel *et al.* at SAC 2009 [8]. It improves the rebound attack by extending the number of controlled rounds from 2 to 3. The idea is to utilize the independence and the freedom of each search procedure as much as possible. As a result, without increasing the time and memory, 3 rounds of the differential path can be fulfilled efficiently.

Super-Sbox analysis for the rebound attack was independently proposed by Lamberger *et al.* at Asiacrypt 2009 and by Gilbert and Peyrin at FSE 2010 [9].

Super-Sbox combines 2 non-linear layers and 1 diffusion layer to 1 non-linear layer with a larger substitution-box. It can extend the inbound phase by one more round. As a side effect, attackers need to spend more time and memory to exploit the differential property of the Super-Sbox.

Peyrin proposed a differential path for ECHO with an increased granularity [26]. This can reduce the number of active bytes inside an active BigWord, and thus the attack complexity can be reduced.

4 Framework of Non-Full-Active Super-Sbox Analysis

In this section, we use the following notations:

r: a number of rows and columns in a state.
c: a number of bits of each cell (word) in a state.
s: a number of non-active columns in the initial state of the differential path.
$Col(x)$: a state where x columns are fully active, namely, $r \times x$ bytes are active.
$SR(Col(x)), SR^{-1}(Col(x))$: a state where $Col(x)$ is passed over SR and SR^{-1}.
F: a state where all bytes are active.
x/y: a state where y bytes become non-active from a state x.

In the Super-Sbox analysis, as long as we follow the strategy of Gilbert and Peyrin [9], the attack complexity is lower-bounded by 2^{rc}. In this section, we present a new framework called non-full-active Super-Sbox analysis which can detect non-ideal properties with a lower complexity. We first make a truncated differential path whose inbound part, in particular inside the Super-Sbox, consists of non-full-active states. For non-active bytes, the differential transition 0 to 0 is always held regardless of its value, and thus attackers can freely choose the value without breaking the path. This gives attackers the freedom degrees to adjust other bytes inside the Super-Sbox.

Non-full-active Super-Sbox analysis can be applied to AES-based permutations. We assume that the MixColumns operation is composed of MDS matrix [5]. Namely, the sum of the number of active bytes in the input and output states is greater than or equal to $r + 1$, otherwise 0.

4.1 Non-Full-Active Truncated Differential Path

We show a generic description of the non-full-active differential path. The differential path has a parameter s, which is the number of non-active columns in the initial state. The parameter s determines the complexity of the attack. The differential path is depicted in Fig. 4 with instantiating the case $r = 8$ and $s = 3$.

To make the differential path, we start from the state after the 2nd and 5th rounds, whose states are $Col(1)/s$ and $SR^{-1}(Col(1))/(r-(s+1))$, respectively. The differential propagation through the 3rd round in forward and the 5th round in backward are deterministic, which result in $F/Col(s)$ and $F/SR^{-1}(Col(r-(s+1)))$, respectively. We then need to check that the differential propagation

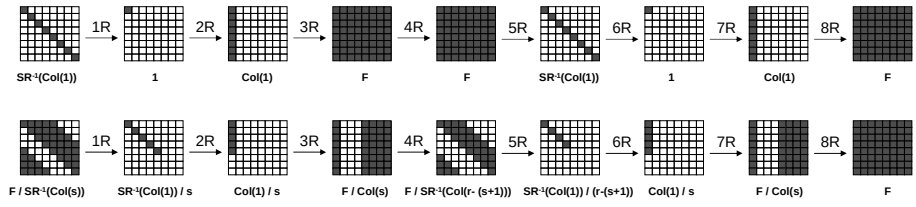

Fig. 4. (Bottom) New differential path for 8-round AES-based permutations with instantiating $r = 8$ and $s = 3$. (Top) Previous path for the Super-Sbox analysis [9].

through the MixColumns operation in the 4th round is consistent with the MDS property. Because input and output states have $r - s$ and $r - (r - (s + 1))$ active bytes in each column respectively, the sum of active bytes in the input and output is $r - s + r - (r - (s + 1)) = r + 1$. Hence, the differential path is consistent with the MDS property. Next, we determine the differential propagation through the 6th round in forward. The number of active bytes should be reduced as much as possible after the 6th round in order to make the target non-ideal property hard for an ideal permutation. Hence, we maximize the number of non-active bytes with satisfying the MDS property, which results in the state $Col(1)/s$. Similarly, we determine the differential propagation through the 2nd round in backward. We make the number of active bytes to be the same as the state after the 6th round[2], which results in $SR^{-1}(Col(1))/s$. The rest of the path is deterministic.

4.2 Low Complexity Inbound Phase

We explain how to compute the inbound phase for our path. Details of states inside the inbound phase are shown in Fig. 5, with denoting each state by $\#i$, where $0 \leq i \leq 8$. The inbound phase starts from the state after the SubBytes in the 3rd round (#0) and the state input to the 6th round (#8). The goal of the inbound phase is finding paired values satisfying the differential path through #0 to #8. We find 2^c such paired values with 2^c computations and 2^c memory.

States #0 and #8 include $r - s$ and $s + 1$ active bytes, respectively. First, we choose and fix the differences of all active bytes in #0 and the differences of s active bytes out of $s + 1$ active bytes in #8. Then, for each 2^c possible differences of the last active byte in #8, we aim to store a corresponding paired value. Due to the linearity of the operations, we can compute the corresponding differences in state #2 and corresponding s-byte differences in each column of #6. The Super-Sbox analysis can be applied between #2 and #6, namely we can compute them column by column independently. Previous Super-Sbox analysis spent 2^{rc} of time and 2^{rc} of memory for this computation, while we efficiently connect these two states by using the freedom degrees of the non-active states.

[2] With a lower probability, the number of active bytes can be smaller. However, this will not lead to any advantage in the distinguishing attack.

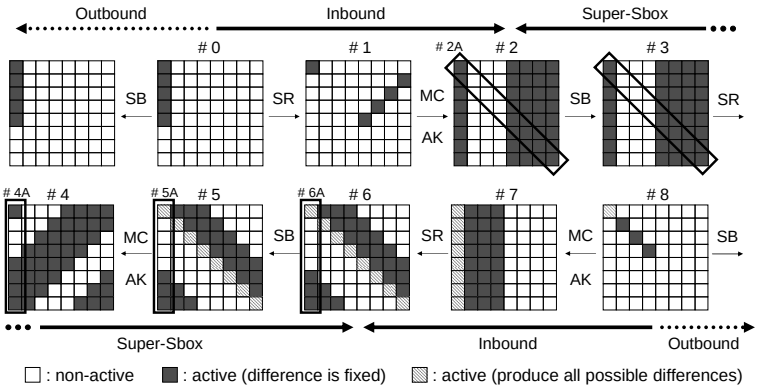

Fig. 5. Inbound phase for the new differential path with non-full-active states.

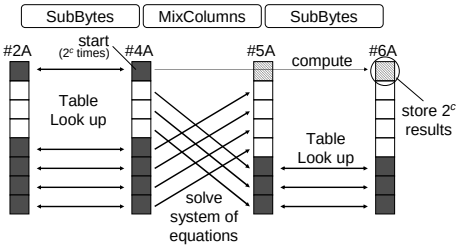

Fig. 6. Computation procedures inside each Super-Sbox.

In the following, we only show the Super-Sbox computations in the left most column, which is emphasized with bold squares in Fig. 5. The other columns can be connected with the same procedure.

Computation procedure inside the Super-Sbox. The operations inside the Super-Sbox are shown in Fig. 6. Because the ShiftRows operation does not give any impact inside the Super-Sbox, we omit it in Fig. 6. To stress that each Super-Sbox is computed column by column, we denote the states inside the Super-Sbox by #2A, #4A, #5A, and #6A in Fig. 6. The goal of this procedure is efficiently producing 2^c paired values which satisfy the fixed part of the differences of #2A and #6A. This procedure finds 2^c paired values with a time complexity of 2^c and 2^c memory. The attack procedure is as follows.

0. For each active byte whose difference is fixed in #2A and #6A, compute SB and Inverse-SB for all possible 2^c values and a fixed difference. Store these 2^c values and corresponding output differences as a look up table. We sort tables according to the output differences so that table look-up only requires 1 memory access. As a result, $(r - s) + s = r$ look-up tables are prepared.

1. Choose a difference of one active byte in #4A. (The top byte of #4A is chosen in Fig. 6.)
2. We have other $r-s-1$ active bytes in #4A and need to make sure the same number of bytes in #5A are non-active. This is done by solving a system of equations and we will obtain one solution of the system. As a result, differences in #4A and #5A become consistent and are uniquely fixed.
3. From a fixed difference of #4A and the given difference of #2A, for each active byte, we obtain a pair of values which connects these differences by looking up tables generated in Step 0. Do the same for fixed s-byte differences of #6A and #5A. Note that values for non-active bytes are not fixed yet.
4. Then we connect the values of active bytes of #4A and #5A. We use the freedom degrees of non-active bytes to effectively achieve this. There are s non-active bytes in #4A and s active bytes in #5A whose values are fixed in Step 3. By solving a system of equations, we can calculate the values of s non-active bytes in #4A so that the fixed s bytes of #5A can be consistent.
5. With the fixed values in #4A, we compute the non-fixed active byte in #5A, and further compute the corresponding value in #6A. We store entire values and differences of states #2A and #6A in a table.
6. We iterate Step 1 to Step 5 2^c times by changing the difference of the chosen active byte in #4A.

Complexity of inbound phase. We assume r and s are enough small compared to 2^c (e.g. $r = 8, s = 3$, and $2^c = 256$ in Fig. 4). Step 0 requires 2^c computations and 2^c memory. Step 1 to Step 4 can be computed with a complexity of 1 (Based on the assumption, the cost for looking-up r tables and solving systems of equations of size s are ignored). Step 5 uses a memory of 1. Because Steps 1 to 5 are repeated 2^c times in Step 6, the complexity of this procedure is time 2^c and memory 2^c. Note that 2^c values and differences of the non-fixed active byte are stored in the table. Therefore, we obtain 1 solution on average for any difference of the non-fixed byte.

After we finish the computation for all Super-Sboxes, we choose a difference of the non-fixed byte in #8 in Fig 5. For each of its possible 2^c differences, we compute the corresponding difference in #6, and obtain the value which connects #2 to #6 by looking up each Super-Sbox. Note, we obtain one solution on average for any pair of differences in #2 and #6. To sum up, we can obtain 2^c starting points, which are solutions of the inbound phase, with time 2^c and memory 2^c. In other words, we obtain a starting point with time 1 on average.

4.3 Outbound phase and the freedom degrees

After the inbound phase, we compute the outbound phase. The differential path described in Fig. 4 has two probabilistic differential propagations: 1) the backward computation through the 2nd round and 2) the forward computation through the 6th round. In both rounds, the MixColumns or InverseMixColumns operations need to produce s non-active bytes. Therefore, for each of these

Table 2. The complexity to find a property with our attack and ideal permutation.

s	1	2	3	4	5	6	7	8
Ours	2^{2c}	2^{4c}	2^{6c}	2^{8c}	2^{10c}	2^{12c}	2^{14c}	2^{16c}
Ideal	$2^{\frac{cr}{2}}$	2^{cr}	$2^{\frac{3cr}{2}}$	2^{2cr}	$2^{\frac{5cr}{2}}$	2^{3cr}	$2^{\frac{7cr}{2}}$	2^{4cr}

rounds, the success probability is 2^{-cs}. Finally, this attack requires 2^{2cs} starting points for the outbound, and each starting point is generated with time 1 on average. Hence, with a time 2^{2cs}, we find a pair following the differential path.

We also need to confirm that the available freedom degree is enough. Our attack starts from the states #0 and #8 in Fig. 5. #0 and #8 include $r-s$ and $s+1$ active bytes respectively, and thus we have $2^{c(r+1)}$ freedom degrees in total. Hence, as long as the parameter s satisfies $2^{c(r+1)} \geq 2^{2cs}$, which is converted to below, we have enough freedom degrees.

$$s \leq \frac{r+1}{2} \tag{1}$$

4.4 Target Class of AES-Based Permutations and an Example

Let us consider the complexity for an ideal permutation. The last MixColumns is not taken into account because it is fully linear. Hence, the problem is regarded as finding a crs-bit collision. A crs-bit collision can be found by the birthday attack because attackers have enough freedom degrees due to Eq. (1). Hence, the complexity for an ideal permutation is $2^{\frac{crs}{2}}$. The comparison of the non-full-active Super-Sbox analysis and the ideal case is shown in Table 2.

From Table 2, we can see $r > 4$ is a condition so that our attack can work. Therefore, our attack cannot be applied to AES ($r = 4$). Note that the ECHO permutation is regarded as an AES-based permutation with $r = 4$ at a BigWord level. However, it has other structures and this enables us to greatly reduce the attack complexity on the ECHO permutation. See Section 5 for details.

Let us consider an application for a real primitive. Grøstl-256 uses an AES-based permutation with $r = c = 8$. In previous Super-Sbox analysis [9], the 8-round permutation is distinguished with time 2^{112} and memory 2^{64}, which is too expensive to be implemented. In our attack, we choose $s = 3$, whose differential path is shown in Fig. 4. Consequently, from Table 2, we can detect a pair of values following the differential path with time 2^{48} and 2^8 memory, while finding a pair of values in an ideal permutation requires 2^{96}, which is infeasible. Choosing other s is also possible as long as $s \leq 4$.

5 Applications to ECHO and Grøstl

5.1 New Observations on ECHO

In this section, we explain several new observations on the ECHO permutation when dealing with the dedicated byte-wise differential path.

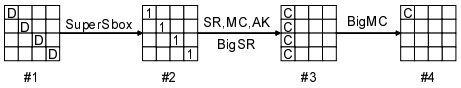

Fig. 7. A differential path for a 1-round ECHO permutation.

Complexity analysis for jointed MixColumns and BigMC. In the ECHO permutation, 2-round AES permutation inside BigSB can be considered as a non-linear layer with Super-Sboxes and a diffusion layer consisting of ShiftRows, MixColumns and AddRoundKey. Note that from the second MixColumns inside BigSB to the following BigMC are successively performed. We show that the linearity of jointed MixColumns and BigMC should be considered to correctly compute the complexity for certain differential paths.

As an example, let us check the complexity for the differential path shown in Fig. 7 assuming the differences and real values at state #1 have full freedom. In the previous analysis [26, Appendix B], the complexity for this differential path is likely to be divided into three parts and analyzed independently. State #1 to #2 can be fulfilled when the output of each active Super-Sbox has only 1 active byte. Since there are totally 12 bytes required to be zero, the probability is regarded as 2^{-96}. The complexity from #2 to #3 is 1. And since 12 bytes are required to be zero from #3 to #4, the probability is regarded as 2^{-96}. As a result, the total probability is regarded as $2^{-96\times 2} = 2^{-192}$. However we show that MixColumns and BigMC cannot be considered separately, and thus the correct probability needs to be reconsidered.

We can see that the freedom of the difference for state #2 or #3 is at most 2^{32}, since #2 has only 4 active bytes. As a contradiction for the previous analysis, the freedom of difference at #3 (2^{32}) seems impossible to fulfill the differential propagation to #4 (2^{-96}). However, we show that this propagation is fulfilled only with a probability of 2^{-24}, and thus 2^{32} freedom degrees are enough.

This fact can be understood from two directions. First, for a position-fixed active byte and the fixed MDS matrix used in MixColumns between #2 and #3, the 4 active bytes inside each active BigWord at #3 has a fixed linear relationship. Then if BigMC generates the required difference at #4 for one of 4 active-byte positions with a probability of 2^{-24} (e.g. 4 top-left bytes from 4 active BigWord at #3 generate 1 active byte at the top-left of state #4), the other three active-byte positions become the same differential pattern at #4 with probability 1. Another interpretation is that one can switch the operation order, namely performing BigMC first and MixColumns later. When 4 active bytes in #2 generate only 1 active byte through BigMC with a probability of 2^{-24}, the differential path from #3 to #4 through MixColumns is fulfilled with probability 1. As a result, the total complexity is $2^{96+24} = 2^{120}$ instead of 2^{192}. Note that this fact was independently discovered by [27] as SuperMixColumns.

Freedom of the differential path inside BigSB. We can use the freedom of the differential path inside BigSB to reduce the attack complexity. Our attacks

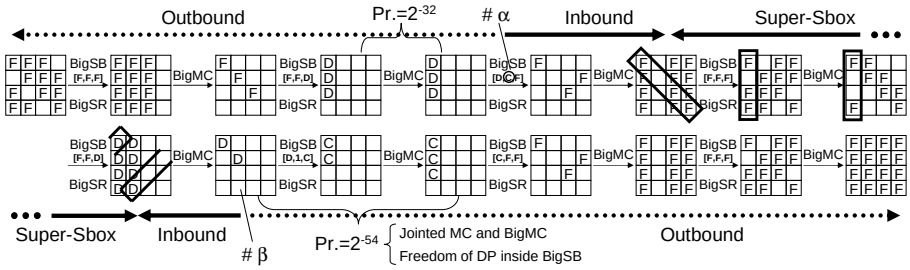

Fig. 8. Non-full active differential path for 8-round ECHO permutation.

only care about the differences at the start and end states of the permutation. We notice that while keeping the differential path at a BigWord level, attackers can use the freedom of the differential paths at a byte level inside BigSB.

We again use the differential path in Fig. 7 as an example. In order to fulfill the differential path, the 4 active bytes in state #2 must be at the same position inside the leftmost column of each BigWord[3]. As a result, the differential path inside the BigSB has 4 choices for the positions of active bytes, and thus, the complexity for the differential path in Fig. 7 can be reduced from 2^{120} to 2^{118}.

5.2 Attack on Full-Round ECHO Permutation

Truncated Differential path. We use the differential path explained in Section 4.1 with parameter $s = 1$ at a BigWord level, which is shown in Fig. 8. We use the notation BigSB$[x, y, z]$, where $x, y, z \in \{F, D, C, 1\}$ to show that x, which is the input differential pattern to BigSB, changes into y after the 1st AES-round and into z after the 2nd AES-round.

Inbound phase. The detailed differential path for the inbound phase is described in Fig. 9. The inbound phase starts from a middle of BigSB in the 3rd round ($\#\alpha$) and the input state to the 6th round ($\#\beta$), where the differential form in $\#\alpha$ is C. We first choose and fix a difference of $\#\alpha$ and a difference of one of active BigWords of $\#\beta$, and compute the corresponding differences of #2 and #6. In the inbound phase, for each of 2^{32} possible differences of the non-fixed active BigWord in $\#\beta$, we find a pair of values that satisfies the chosen differences of $\#\alpha$ and $\#\beta$. The attack procedure follows the one explained in Section 4.2 with some optimization specific to ECHO. In the followings, we describe details to compute 1 Super-Sbox of ECHO with the size of 128 bits.

0. Generate a look-up table for each 4 active BigWord with fixed difference in #2A and #6A, With the procedure in Section 4.2, this costs 2^{128} time

[3] If 4 active bytes in state #2 are in different positions inside each BigWord, the path for Fig. 7 becomes impossible. This may be used as a countermeasure of our attack.

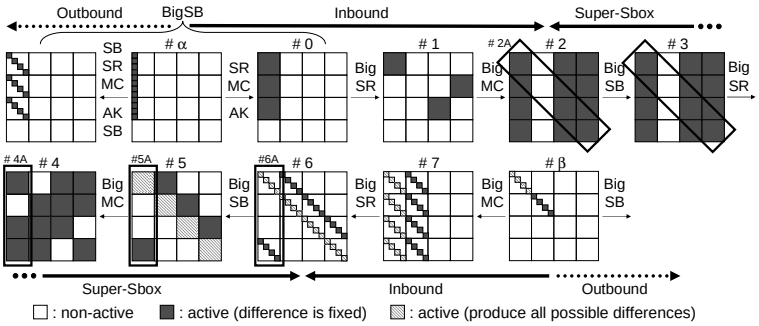

Fig. 9. Inbound phase for 8-round ECHO permutation.

and 2^{128} memory. [8] pointed out that this could be performed efficiently by looking inside BigSB. The BigSB can be regarded as 4 Super-Sboxes (SB, SR, MC, AK, SB) with the size of 32 bits and the linear part (SR, MC, AK). Then, for a given output difference of BigSB, we can calculate back the corresponding difference of the linear part, and thus values are searched by looking up four 32-bit Super-Sboxes independently. Hence, look-up tables for 4 BigWords can be generated by computing 16 Super-Sboxes, which requires 16×2^{32} in both time and memory.

1. Choose a difference of one active BigWord in #4A.
2. By solving a system of equations, compute differences of 2 active BigWords in #4A so that 2 target BigWords in #5A can be non-active.
3. For each active BigWord with fixed difference, obtain a pair of values which connects differences between #4A and #2A, and between #6A and #5A by looking up tables generated in Step 0.
4. By solving a system of equations, calculate the value of 1 non-active BigWord in #4A so that the fixed value of 1 BigWord in #5A can be consistent.
5. With the fixed paired values in #4A, compute the non-fixed active BigWord in #5A and #6A. Only if the computed difference of #6A has the diagonal form D, store entire values and differences of states #2A and #6A in a table.
6. Iterate Steps 1 to 5 2^{128} times by changing the difference of the chosen BigWord in #4A.

In Step 0, look up tables are generated with 2^{36} time and 2^{36} memory. Steps 1 to 5 are iterated 2^{128} times. In Step 5, the computed difference has the diagonal form D with a probability of $2^{32}/2^{128} = 2^{-96}$, and thus we store 2^{32} data after 2^{128} iterations. Hence, the complexity for 1 Super-Sbox with the size of 128 bits is 2^{128} computations and $2^{36}+2^{32}$ memory. Note that we need $2^{36}+(4\times 2^{32}) < 2^{37}$ memory for 4 Super-Sboxes. In the end, the inbound phase generates 2^{32} starting points with 2^{128} computations and 2^{37} memory, which is 2^{96} computations on average to generate 1 starting point.

Success probability and freedom degrees. If details are considered, Step 3 succeeds only probabilistically. Look-up tables for each BigWord consists of 4 Super-Sboxes with the size of 32 bits. Assume that each Super-Sbox has the same property as the AES Sbox. Namely, for a randomly given a pair of input and output differences, with a probability of approximately 2^{-1}, there exists approximately 2 paired values satisfying the differences. In Step 3, we look-up 16 Super-Sboxes. Hence, the success probability is 2^{-16} and we obtain 2^{16} paired values. We compute Steps 4 and 5 for all 2^{16} paired values, and thus they are computed 2^{128} times in total by the 2^{128} iteration of Step 6. Consequently, the total time and memory for the inbound phase will not change. Note that the estimation by using average numbers is imprecise only if the cost for the outbound phase is cheaper than the inbound phase. Because our attack iterates the inbound phase 2^{54} times, the evaluation with average numbers is valid.

We then check the freedom degrees. In the inbound phase, we can choose up to 2^{96} differences for $\#\alpha$ and 2^{32} differences for the fixed active BigWord in $\#\beta$. Hence, the inbound phase can be iterated 2^{128} times and thus we can generate 2^{160} starting points in maximum, which are enough to satisfy the outbound.

Outbound phase. The differential path shown in Fig. 8 includes two probabilistic differential propagations.

InverseBigMC in the 2nd round. For each of diagonal positions, Inverse MixColumns outputs one non-active byte. This probability is $(2^{-8})^4 = 2^{-32}$.

BigSB and BigMC in the 6th round. Observations explained in Section 5.1 are applied for this part. The probability that the differences in 2 BigWords propagate as $D \rightarrow 1 \rightarrow C$ is $(2^{-24})^2 = 2^{-48}$. By taking the freedom of the differential path inside BigSB into account, the probability becomes $4 \times 2^{-48} = 2^{-46}$. In the BigMC operation, MC is computed for 4 positions. Due to the property of jointed MixColumns and BigMC operations, all of the 4 positions will make 1 non-active byte with a probability of 2^{-8} in total. As a result, the total success probability of the 6th round is $2^{-46} \times 2^{-8} = 2^{-54}$.

In the end, the success probability of the outbound phase is $2^{-32} \times 2^{-54} = 2^{-86}$.

Total complexity and comparison with ideal case. In our attack, we generate 2^{86} starting points and each of them is generated with 2^{96} computations on average. Hence, the total complexity is $2^{86} \times 2^{96} = 2^{182}$. Note that this attack requires 2^{37} memory. On the other hand, for the ideal case, the property is regarded as finding a 512-bit collision. This requires 2^{256}, which is much higher than our attack on ECHO.

5.3 Improving Semi-Free-Start Collisions on 7-round Grøstl-512

We improve the semi-free-start collision attack on 7-round Grøstl-512 compression function proposed by Mendel *et al.* [17]. It uses the previous Super-Sbox

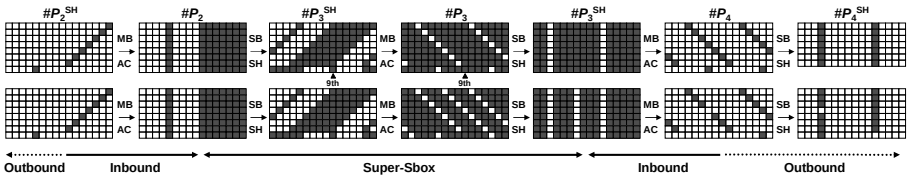

Fig. 10. (Bottom) New differential path for Grøstl-512. (Top) Previous path.

analysis and thus requires 2^{64} memory. We show the memory can be reduced to 2^{56} with the non-full-active Super-Sbox analysis. Because our outbound phase is the same as [17], we only explain the inbound phase.

In the Super-Sbox analysis with a rectangle state such as $r \times 2r$, several Super-Sboxes include non-active bytes. Hence, the framework explained in Section 4 can be applied and the data stored for each Super-Sbox can be reduced. In the previous differential path [17, Fig.7] shown in Fig. 10, the 9th Super-Sbox at $\#P_3^{SH}$ takes a full-active column as input and output a full-active column, which requires 2^{64} memory. In fact, this is a bottleneck in the entire attack.

We reduce the number of active bytes where we choose the differences at the initial step of the inbound phase ($\#P_4$). This results in a differential path where each Super-Sbox has at least one non-active byte. The new path is shown in Fig. 10. Each Super-Sbox can be computed based on the procedure explained in Section 4.2, which results in generating 2^{56} starting points with 2^{56} time and 2^{56} memory. Note that the differential propagation from $\#P_3^{SH}$ to $\#P_3$ must be consistent with the MDS property. We confirmed that the amount of memory could not be below 2^{56} due to this limitation.

Because we reduced the number of active bytes, the freedom degree was also reduced. The success probability of the outbound phase is 2^{-152}, and thus we need 2^{152} starting points. Because our attack can choose 22-byte differences (8-byte for $\#P_2^{SH}$ and 14-byte for $\#P_4$) at the initial step, up to $2^{8\times 22} = 2^{176}$ starting points can be produced, which is enough to satisfy the outbound path.

6 Conclusions

We presented the non-full-active Super-Sbox analysis which can detect non-ideal properties of a class of AES-based permutations with a low complexity. The core idea is using a differential path consisting of only non-full-active states. This gives us the freedom to efficiently control inside the Super-Sbox. We then applied this framework to the full-round ECHO permutation by taking properties specific to ECHO into account. Consequently, our attack could detect a non-ideal property with time 2^{182} and memory 2^{37}. Note because of the convolution operation, our attack cannot be extended to the hash or compression function. We then applied our approach to Grøstl to obtain the distinguishing attack on the 8-round Grøstl-

256 permutation with a practical cost, and to obtain an improvement on the semi-free-start collision attack on the 7-round Grøstl-512 compression function.

References

1. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: EUROCRYPT 2005. Volume 3494 of LNCS. Springer-Verlag (2005) 19–35
2. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: CRYPTO 2005. Volume 3621 of LNCS., Springer-Verlag (2005) 17–36
3. U.S. Department of Commerce, National Institute of Standards and Technology: Federal Register /Vol. 72, No. 212/Friday, November 2, 2007/Notices. (2007)
4. U.S. Department of Commerce, National Institute of Standards and Technology: Specification for the ADVANCED ENCRYPTION STANDARD (AES) (Federal Information Processing Standards Publication 197). (2001)
5. Daemen, J., Rijmen, V.: The design of Rijndael: AES – the Advanced Encryption Standard (AES). Sprnger-Verlag (2002)
6. Peyrin, T.: Cryptanalysis of Grindahl. In: ASIACRYPT 2007. Volume 4833 of LNCS., Springer-Verlag (2007) 551–567
7. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In: FSE 2009. Volume 5665 of LNCS., Springer-Verlag (2009) 260–276
8. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In: SAC 2009. Volume 5867 of LNCS., Springer-Verlag (2009) 16–35
9. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In: FSE 2010. Volume 6147 of LNCS. (2010) 365–383
10. Rijmen, V., Toz, D., Varici, K.: Rebound attack on reduced-round versions of JH. In: FSE 2010. Volume 6147 of LNCS. (2010) 286–303
11. Wu, S., Feng, D., Wu, W.: Cryptanalysis of the LANE hash function. In: SAC 2009. Volume 5867 of LNCS., Springer-Verlag (2009) 126–140
12. Wu, S., Feng, D., Wu, W.: Practical rebound attack on 12-round Cheetah-256. In: ICISC 2009. Volume 5984 of LNCS., Springer-Verlag (2010) 300–314
13. Matusiewicz, K., Naya-Plasencia, M., Nikolić, I., Sasaki, Y., Schläffer, M.: Rebound attack on the full LANE compression function. In: ASIACRYPT 2009. Volume 5912 of LNCS., Springer-Verlag (2009) 106–125
14. Mendel, F., Rechberger, C., Schläffer, M.: Cryptanalysis of Twister. In: ACNS 2009. Volume 5536 of LNCS., Springer-Verlag (2009) 342–353
15. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full Whirlpool compression function. In: ASIACRYPT 2009. Volume 5912 of LNCS., Springer-Verlag (2009) 126–143
16. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: The rebound attack and subspace distinguishers: Application to Whirlpool. Cryptology ePrint Archive, Report 2010/198 (2010)
17. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Rebound attack on the reduced Grøstl hash function. In: CT-RSA 2010. Volume 5985 of LNCS., Springer-Verlag (2010) 350–365
18. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl addendum. Submission to NIST (updated) (2009)

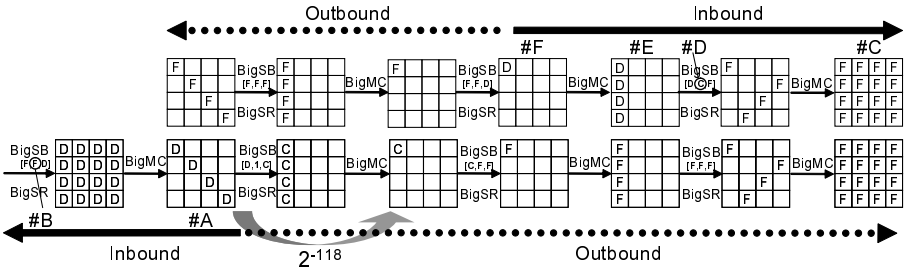

Fig. 11. Differential path for 7-round ECHO permutation.

19. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 proposal: ECHO. Submission to NIST (updated) (2009)
20. Wu, H.: The hash function JH. Submission to NIST (updated) (2009)
21. Khovratovich, D., Biryukov, A., Nikolić, I.: The hash function Cheetah: Specification and supporting documentation. Submission to NIST (2008)
22. Indesteege, S.: The LANE hash function. Submission to NIST (2008)
23. Ewan Fleischmann, C.F., Gorski, M.: The Twister hash function family. Submission to NIST (2008)
24. Rijmen, V., Barreto, P.S.L.M.: The Whirlpool hashing function. Submitted to NISSIE (2000)
25. Peyrin, T.: Improved differential attacks for ECHO and Grøstl. In: CRYPTO 2010. Volume 6223 of LNCS. (2010) 370–392
26. Peyrin, T.: Improved differential attacks for ECHO and Grøstl. Cryptology ePrint Archive, Report 2010/223 (2010) Extended version of the CRYPTO 2010 article.
27. Schläffer, M.: Subspace distinguisher for 5/8 rounds of the ECHO-256 hash function. In: Preproceedings of SAC 2010. (2010) 379–398
28. Ideguchi, K., Tischhauser, E., Preneel, B.: Improved collision attacks on the reduced-round Grøstl hash function. Cryptology ePrint Archive, Report 2010/375 (2010) Appeared in the accepted papers list of ISC 2010.

A Attack Procedures on 7-Round ECHO Permutation

Using the differential path shown in Fig. 11, we present an attack on the 7-round ECHO permutation with time of 2^{118} and memory of 2^{38}.

Step 1 An attacker picks up a difference at state #A (from 2^{128} patterns) and calculates the difference back to #B (state after the second SubBytes).

Step 2 The transformation from #B to #C can be divided into 64 independent 4-byte Super-Sboxes. For each Super-Sbox with fixed output difference, by testing all 2^{32} output values, the attacker can make a table of all possible input values and differences. At the end of Step 2, all the possible pairs at #C are stored in a table named T1 that is composed of 64 small tables each with size 2^{32}. Hence, we need 2^{38} memory for this step.

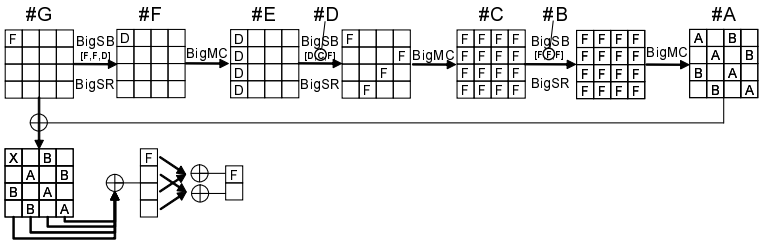

Fig. 12. Differential path of 3-round single-pipe ECHO compression function.

Step 3 For each active BigWord at #D, the attacker picks up a difference and calculates a corresponding difference of BigColumn at #C. Then attacker checks whether the calculated difference exists in T1. Once it exists, the attacker uses the corresponding real values at #C to calculate back the real values at #D. This test is repeated for all possible differences for each active BigWord of #D, and all possible differences and real values at #D are stored in a new table named T2. The time and memory for Step 3 are both 2^{32}.

Step 4 For all the possible pairs at each active BigWord at #D, the attacker calculates the pairs at #E and stores the results as a table named T3.

Step 5 For all the possible 2^{32} differences at #F, the attacker calculates the differences at #E and checks whether the calculated difference exists in T3.

When Steps 1 to 5 are applied to Fig. 11, the inbound and backward outbound phases are merged and calculated efficiently. ABy applying the procedure once, with time of 2^{32} and memory of 2^{38}, the attacker gets 2^{32} start points. Note that with the 2^{128} freedom of the differences at #A, the forward outbound phase can be fulfilled. As a result, the total complexity is 2^{118} in time by the observations in Section 5.1 and 2^{38} in memory.

B Attack on 3-Round ECHO-SP Compression Function

Note that, for the attack in Appendix A, there is no specific requirement for the differences at state #A. Using this property we can find a non-ideal property of the 3-round single-pipe ECHO compression function specified in [19].

The differential path is shown in Fig. 12. An attacker makes sure the differences at #A can be cancelled in the compression calculation, *i.e.* for each row of BigWords at #A, the difference labeled as A is the same with the one labeled as B. By applying the procedure in Appendix A, this differential path can be satisfied using 2^{32} time and 2^{38} memory, while it costs 2^{64} for the ideal case.